strlcpy() and strlcat()

Daniel Plakosh, Software Engineering Institute [vita[1]]

2006-01-30; Updated 2008-10-06 L3 / D/P, L[2]

The standard C library includes functions that are designed to prevent buffer overflows, particularly `strncpy()` and `strncat()`. These universally available functions discard data larger than the specified length, regardless of whether it fits into the buffer. These functions are deprecated for new Windows code because they are frequently used incorrectly.

Development Context

Copying and concatenating character strings

Technology Context

C, UNIX, FreeBSD, OpenBSD, NetBSD, MacOS X, Solaris

Attacks

Attacker executes arbitrary code on machine with permissions of compromised process or changes the behavior of the program.

Risk

Improper use of the `strncpy()` and `strncat()` functions can result in buffer overflow vulnerabilities.

Description

The `strlcpy()` and `strlcat()` functions copy and concatenate strings in a less error-prone manner than the corresponding C99 functions. These functions' prototypes are as follows:

```
size_t strlcpy(char *dst, const char *src, size_t size);
```

```
size_t strlcat(char *dst, const char *src, size_t size);
```

The `strlcpy()` function copies the null-terminated string from src to dst (up to `size` characters). The `strlcat()` function appends the null-terminated string src to the end of dst (but no more than `size` characters will be in the destination).

To help prevent writing outside the bounds of the array, the `strlcpy()` and `strlcat()` functions accept the full size of the destination string as a size parameter. For static buffers, this value is easily computed at compile time using the `sizeof()` operator.

Both functions guarantee that the destination string is null terminated for all nonzero-length buffers.

The `strlcpy()` and `strlcat()` functions return the total length of the string they tried to create. For `strlcpy()` that is simply the length of the source; for `strlcat()` it is the length of the destination (before concatenation) plus the length of the source. To check for truncation, the programmer needs to verify that the return value is less than the size parameter. If the resulting string is truncated, the programmer now has the number of bytes needed to store the entire string and may reallocate and recopy.

1. http://buildsecurityin.us-cert.gov/bsi/about_us/authors/268-BSI.html (Plakosh, Daniel)

Neither `strlcpy()` nor `strlcat()` zero-fill their destination strings (other than the compulsory null byte to terminate the string). This results in performance close to that of `strcpy()` and much better than `strncpy()` [Miller 99].

Unfortunately, `strlcpy()` and `strlcat()` are not universally available in the standard libraries of UNIX systems. Both functions are defined in `string.h` for many UNIX variants, including Solaris, but not for GNU/Linux. Because these are relatively small functions, however, you can easily include them in your own program's source whenever the underlying system doesn't provide them. It is still possible (however unlikely) that the incorrect use of these functions will result in a buffer overflow if the specified buffer size is longer than the actual buffer length.

References

[Miller 99] Miller, T. C. & de Raadt, T. "strlcpy and strlcat—Consistent, Safe String Copy and Concatenation," 175-178. *Proceedings of the FREENIX Track, 1999 USENIX Annual Technical Conference*. Monterey, CA, June 6-11, 1999. Berkeley, CA: USENIX Association, 1999. http://www.usenix.org/publications/library/proceedings/usenix99/full_papers/millert/millert.pdf.

Pearson Education, Inc. Copyright